Fuzzy ID3: Evaluation, Rule Generation and Network Mapping

By

Kishori Mohan Konwar

Dr. Sushmita Mitra
Associate Professor

Professor Sankar K. Pal
Distinguish Scientist and Head

Machine Intelligence Unit
Indian Statistical Institute
Calcutta 700 035

Certificate of Apporval

This is to certify that this dissertation titled **Fuzzzy ID3: Evaluation, Rule Generation and Network Mapping** submitted by **Kishori Mohan Konwar** towards partial fulfillment of the requirements for the degree of *M.Tech. (Computer Science)* at the Indian Statistical Institute, Calcutta, embodies the work carried out under our supervision. His work is satisfactory.

Dr. S. Mitra

Date:

Professor S. K. Pal

Date:

Acknowledgement

I am deeply indebted to the Machine Intelligence Unit, Indian Statistical Institute, Calcutta and to my advisors, Professor Sankar K. Pal and Professor Sushmita Mitra, for their constant encouragement and valuable guidance throughout this dissertation work.

I would like to thank Anil Kumar Ghosh, Sanjeev Kumar Mishra and Tapas Chakraborty for helping to use Latex efficiently while preparing this report.

Finally but immensely, thanks are due to Sandip Das for many of his suggestions which made the implementation of artificial neural network, in the last part of my dissertation, lot easier.

Kishori Mohan Konwar — 25 July,2000

M.Tech.(Computer Science) — ISI,Calcutta.

Contents

Abstract

A fuzzy knowledge-based neural network is generated using rules extracted from fuzzy ID3. The fuzzy decision tree can handle numeric features and/or linguistic features and class membership values, while fuzzy entropy is used at the node level. A new measure is developed to evaluate the goodness of the decision tree. Linguistic rules are extracted and quantitatively evaluated. An optimal fuzzy knowledge-based network is automatically generated using the extracted rules. The effectiveness of the system, in terms of recognition score, performance of rules and size of network, is demonstrated on a vowel recognition problem.

Keywords : *Fuzzy ID3, rule generation, classification, rule evaluation, network mapping, neuro-fuzzy computing.*

Chapter 1

Introduction

One of the most significant methods in the area of pattern recognition and classification is the delta rule, which is used for the back-propagation in the training of multilayer perceptrons [1]. However, there is no standard way of generating such a neural network architecture and it is mostly relied upon empirical methods to find a suitable neural network architecture. In the generation of a neural network architecture two fundamental questions arise

- What is the number of nodes required in a hidden layer ?
- What is the number of hidden layers required ?

To help in determining a feedforward neural network architecture, it was shown that a four-layered network with two hidden layers can solve arbitrary classification problems [2]. Irie and Miyake [3] proved that a three-layered back-propagation network with an infinite number of nodes in hidden layer can also solve arbitrary mapping problems.There exist several approaches for dynamically generating the network architecture using growing and/or pruning of nodes and links. However, the determination of an optimal feedforward neural network architecture remains an important research area [4, 5]. We directed our studies to incorporate the aspect of maximum information gain in the generation of a neural network architecture.

Using information entropy is common in machine learning algorithms. One such efficient machine learning for classifying symbolic or non- numeric patterns is ID3 (**Interactive Dichotomizer 3**) [6]. The ID3 algorithm dynamically generates a decision tree using information theoretic concepts. Studies of the ID3 algorithm and back-propagation neural networks revealed strong evidence that ideas similar to the ID3 algorithm may be used to answer the above two fundamental questions concerning the generation of optimal neural

network architecture [7, 8] . Conventional ID3 can be very effective under certain conditions, but has a few serious drawbacks, *viz.*,

- it handles categorical or non-numeric or symbolic attributes only,
- it cannot deal with partial information and
- its efficiency is less for overlapping classes.

Schemes have been proposed in literature [9] to partition the input features into intervals determining appropriate thresholds. Generally artificial neural nets (ANNs) consider a fixed topology of neurons connected by links in a pre-defined manner. These connection weights are usually initialized by small random values. Recently, there have been some attempts in improving the efficiency of neural computation by using knowledge-based nets. This helps in reducing the searching space and time while the network traces the optimal solution. Knowledge-based networks [10, 11, 12] constitute a special class of ANNs that consider crude domain knowledge to generate the initial network architecture, which is later refined in the presence of training data. Such a model has the capability of outperforming a standard MLP as well as other related algorithms including symbolic and numerical ones [10, 11].

We have designed a fuzzy ID3 algorithm capable of handling numeric and/or linguistic attributes.the linguistic partitions along each feature are automatically determined depending on the distribution of pattern points in the feature space. The concept of fuzzy entropy is used at the node level to determine the tree structure. Pruning is used to minimize noise, resulting in a smaller decision with more efficient classification. A new measure is developed to judge the performance of the tree both in terms of performance and size. A smaller tree leads to the generation of more meaningful linguistic rules.

The rules are evaluated using quantitative measures. A new measure to estimate the coverage of these rules is defined. The extracted rules, representing domain knowledge, are mapped to an MLP. This leads to the automatic generation of a fuzzy knowledge-based network. Different mapping schemes are developed and compared. The proposed algorithm leads to a smaller and more efficient system. The effectiveness of the model is demonstrated on a speech recognition problem.

Chapter 2

ID3 and Incorporation of Fuzziness

This chapter begins with the conventional ID3 algorithm which suffers from the drawbacks of

- infeasibility of handling non-numeric data and
- inefficiency in classifying overlapping classes.

To overcome these shortcomings a few alternate fuzzy ID3 algorithms have been proposed and tested on vowel data during this dissertation work. This chapter discusses these algorithms incorporating fuzzines at the input, output and node levels. A new metric, named *T-measure*, is designed to evaluate the efficiency of the decision tree. The chapter concludes with the results obtained from all the above algorithms.

2.1 ID3 algorithm

The ID3 [13] approach to classification consists of a procedure for synthesizing an efficient discrimination tree for classifying pattern that have non-numeric values. The ID3 approach makes use of labeled data and determines how features might be examined in sequence until all the labeled examples have been classified properly. If the data set used to construct the tree is a representative of the much larger ensemble of patterns comprising the original body of the data, then we expect a very large gain through the use of ID3.

An important property of these algorithms is that they implicitly attempt to minimize the size of the tree while optimizing some local quality measure – such as entropy or information content.

The tree is constructed in a data driven manner using depth first. The information content is measured according to [14].

$$Entropy \;=\; -\sum_{i=1}^{l} p_i \log p_i$$

where l is the set of decisions, and p_i is the probability that a training sample in the node represents class i.

Algorithm 2.1: ID3 Algorithm

In the training set there are N patterns from classes $C_i, i = 1, 2, \ldots, l$. Each class C_i has N_i patterns and each pattern has n attributes.

1. *Calculate initial entropy.*

Initial entropy:

$$Ent(I) \;=\; -\sum_{k=1}^{l} \frac{N_k}{N} \log_2 \frac{N_k}{N} = \; -\sum_{k=1}^{l} p_k \log_2 p_i \qquad (2.1.1)$$

where p_i is the *a priori* probability of the ith class.

2. *Select a feature to serve as the root node of the decision tree.*

2.1 For each feature $f_i, i = 1, 2, \ldots, n$, partition the N patterns into two according to their values (0 or 1).

2.2 For any branch population n_{ij} (i for pattern number, j for left or right branch), the number of patterns belonging to class C_k is $n_{ij}(k)$.

Entropy of branch j:

$$Ent(I, f_i, j) \;=\; -\sum_{k=1}^{l} \frac{n_{ij}(k)}{n_{ij}} \log_2 \frac{n_{ij}(k)}{n_{ij}}$$

The entropy of the system after testing on attribute f_i:

$$Ent(I, f_i) = \sum_{j=1}^{2} \frac{n_{ij}}{\sum_j n_{ij}} Ent(I, f_i, j).$$

2.3 Change in entropy after testing on feature f_i:

$$\Delta Ent(f_i) \;=\; Ent(i) - Ent(I, f_i) \qquad (2.1.2)$$

2.4 Select a feature f_k such that

$$\Delta Ent(f_k) > Ent(f_i) \tag{2.1.3}$$

for all $i = 1, 2, \ldots, l, i \neq k$.

2.5 Feature f_k is then the root of the decision tree.

3. *Build the next level of the decision tree.*

Select a feature from the remaining ones, for which the change in entropy is maximum, as the node of the next level.

4. Repeat steps 1 through 3 until either all features are exhausted or sub-populations vanish or change in entropy is zero.

2.2 Fuzzy Sets

In this section we provide some facts related to the *fuzzy sets* and membership functions. The concept of fuzzy sets was introduced by Prof. L. A. Zadeh [15], to quantify vague human perception in terms of mathematical framework in the classical set theory (which is also known as *crisp set*) an element may or may not belong to a particular set but in the *fuzzy* domain each element has an associated *membership* value which indicates the degree of belongingness of an element in that fuzzy set. Formal definition is as follows :

DEFINITION 2.2.1. *A* **fuzzy set** $\mathcal{A}$ *in a space of points* $\mathcal{R} = \{r\}$ *is a class of events with continuum of grades of membership and is characterized by a* **membership function** $\mu_A(r)$ *that associates with each element in* $\mathcal{R}$ *a real number in the interval [0,1] with the value of* $\mu_A(r)$ *at* r *representing the grade of membership of* r *in* $\mathcal{A}$ *. Precisely , a fuzzy set* $\mathcal{A}$ *with its finite number of supports* $r_1, r_2, ..., r_n$ *is defined as a collection of ordered pairs*

$$\begin{aligned} A &= (\mu_A(r_i), r_i), i = 1, 2, \ldots, n \\ &= (\mu_A(r_i)/r_i), i = 1, 2, \ldots, n \end{aligned}$$

where the ***support*** *of A is an ordinary* ***subset*** *of R and is defined as*

$$S(A) = \{\tau | \tau \in R \text{ and } \mu_A(\tau) > 0\}.$$

Here $\mu_i(\tau_i)$, the grade of membership of τ_i in $\mathcal{A}$, denotes the degree to which an event τ_i may be a member of $\mathcal{A}$ or belong to $\mathcal{A}$. For a crisp set

$$\mu_i(\tau_i) = \begin{cases} 1 \text{ if } \tau_i \in A \\ 0 \text{ if } \tau_i \notin A. \end{cases}$$

In pattern classification problems we often need to represent a class with fuzzy boundary in terms of a function. Membership functions can be triangular or trapezoidal representation. One can also use standard S and π functions [16].Each function can be defined in terms of a centre and a crossover point, with membership values 1 or 0.5 respectively.

2.3 Fuzzy ID3

In order to handle numeric or continuous attribute values, some investigations have been made. These include them Fuzzy ID3 [17], RID3 [18], Neuro-Fuzzy [17]. Here we present some fuzzy versions of ID3, focusing mainly on handling linguistic attributes and providing immunity to noise. This is done by

- calculating the input fuzzy membership using *partition values* and
- pruning the decision tree.

As we have already discussed, ID3 algorithm for generating decision trees requires discrete valued attributes. So, if the patterns are of continuous valued attributes then there must be a way to discretize them. We propose a way to do so in terms of linguistic variables *viz.*, *low, medium and high*. This is achieved in two steps,

- the first step involves in calculating the input membership values of each feature into the fuzzy sets *low, medium and high*. Thus, each $n - dimensional$ pattern $F_i = (f_1, f_2, \ldots, f_n)$ is converted into a 3n-dimensional feature

$$F_i = (\mu_{low}(f_1), \mu_{medium}(f_1), \ldots, \mu_{high}(f_n))$$

- The second step is converting each membership value into binary values by considering an appropriate threshold, t.

The algorithm for discretizing the continuous valued patterns is given below. Here we partition each feature into three triangular partitions using only two parameters. We do not consider the arithmetic mean as in [19], but use *quantiles* or *partition values*[1] instead, in order to avoid considering noise patterns. Note that one of the strongest drawbacks of arithmetic mean is that it is very much sensitive to large values (noise). The membership values of the linguistic variables *low, medium and high* is represented in Fig.2.1.

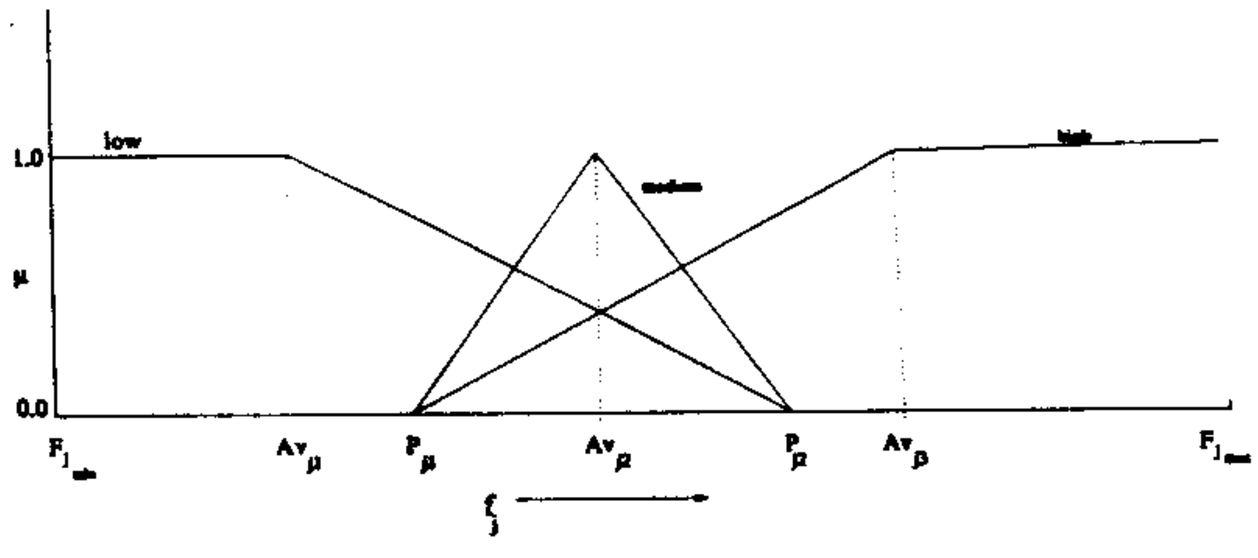

Figure 2.1: Membership Function.

Algorithm 2.2: Algorithm for discretizing continuous features

for each attribute f_i do

begin

$$
\begin{aligned}
f_{j_{min}} &= \min\{f_{1j}, f_{2j}, \ldots, f_{Nj}\} \\
P_{j1} &= first\ partition\{f_{1j}, f_{2j}, \ldots, f_{Nj}\} \\
P_{j2} &= second\ partition\{f_{1j}, f_{2j}, \ldots, f_{Nj}\} \\
f_{j_{max}} &= max\{f_{1j}, f_{2j}, \ldots, f_{Nj}\} \\
Av_{j1} &= \frac{f_{j_{min}} + P_{j1}}{2} \\
Av_{j2} &= \frac{P_{j1} + P_{j2}}{2} \\
Av_{j3} &= \frac{P_{j2} + f_{j_{max}}}{2}
\end{aligned}
$$

[1]Quantiles or partition values are the values of a variate which divide the total frequency into a number of equal parts.

(N is the total number of patterns in the training set)

end

for each pattern F_i **do**

begin

for each feature f_{ij} **do**

begin

$$\mu_{low_j}(F_i) = \begin{cases} 1 & \text{for } f_{ij} < Av_{j1} \\ \frac{P_{j2}-f_{ij}}{P_{j2}-Av_{j1}} & \text{for } Av_{j1} \leq f_{ij} < P_{j2} \\ 0 & \text{otherwise,} \end{cases} \tag{2.3.4}$$

$$\mu_{medium_j}(F_i) = \begin{cases} 0 & \text{for } f_{ij} < P_{j1} \\ \frac{Av_{j2}-f_{ij}}{Av_{j2}-P_{j1}} & \text{for } P_{j1} \leq f_{ij} < Av_{j2} \\ \frac{P_{j2}-f_{ij}}{P_{j2}-Av_{j2}} & \text{for } Av_{j2} \leq f_{ij} < P_{j2} \\ 0 & \text{otherwise,} \end{cases} \tag{2.3.5}$$

$$\mu_{high_j}(F_i) = \begin{cases} 0 & \text{for } f_{ij} < P_{j1} \\ \frac{f_{ij}-P_{j1}}{Av_{j3}-P_{j1}} & \text{for } P_{j1} \leq f_{ij} < Av_{j3} \\ 1 & \text{otherwise.} \end{cases} \tag{2.3.6}$$

For each linguistic feature in the above equations

$$F_{ijk} = \begin{cases} 0 & \text{if } \mu_{low_j}(F_i) < T_j \text{ or } \mu_{medium_j}(F_j) < T_j \text{ or } \mu_{high_j}(F_j) < T_j \\ 1 & \text{otherwise} \end{cases} \tag{2.3.7}$$

for $(m = 1, 2, 3)$ corresponding to *low, medium and high.*

end

end.

Now we focus on the problem of improving the ID3 algorithm to classify the overlapping pattern classes. In ID3 algorithm the sample space is partitioned to form a decision tree. When two sample points from two overlapping classes lie in the intersecting regions the corresponding attributes for these samples are expected to be the same. This implies that both the samples travel through the same path to the same node. Further splitting is not possible since ΔEnt will be zero, which is a stopping condition for constructing the tree. Thus, ID3 fails in an overlapping regions.

In order to achieve further resolution we take the help of membership values of each pattern into each of the classes. These values reflect the relative proximity of the patterns to each of these classes. We assume that the points nearer to the class centers have higher chance of belonging to that class or cluster. So, our membership function is based on distance of a pattern under consideration from the cluster center. Following is an algorithm for this [20].

Consider an l-class problem domain. The membership of the ith pattern in class k, lying in the range $[0, 1]$, is defined as [21, 22]

$$\mu_{ik}(\mathbf{F}_i) = \frac{1}{1 + (\frac{z_{ik}}{f_d})^{f_e}} \quad , \tag{2.3.8}$$

where z_{ik} is the weighted distance of the training pattern $\mathbf{F}_i$ from class C_k, and the positive constants f_d and f_e are the denominational and exponential fuzzy generators controlling the amount of fuzziness in the class membership set.

Fuzziness is incorporated into the ID3 algorithm at the node level by modifying the decision function with classical Shannon's entropy by the inclusion of fuzzy entropy functions. The fuzzy entropy measure considers the membership of pattern to a class and helps enhancing the discriminative power of an attribute. The fuzzy ID3 algorithm for constructing the decision tree is provided below. The salient features of this algorithm are as follows.

- The input membership is calculated in a way to eliminate noise by using *partition values*;
- different entropy functions are investigated to give proper weightage to the classical entropy and fuzzy entropy in order to construct good decision trees; and

- while constructing the tree some nodes are pruned by thresholding on the fraction of patterns reaching that node in order to reduce noise.

Algorithm 2.3: The Fuzzy ID3 Algorithm

In the training set there are N patterns from classes $C_i, i = 1, 2, \ldots, l$. Each class C_i has N_i patterns and each pattern has n attributes.

1. The features of the training patterns are discretized using Algorithm 2.2.

2. Calculate the class memberships of the patterns using eqn(2.3.8).

3. *Calculate initial entropy* $Ent(I)$.

4. *Select a feature to serve as the root node of the decision tree.*

4.1 For each feature $f_i, i = 1, 2, \ldots, n$, partition the N patterns into two according to their values (0 or 1).

4.2 For any branch population n_{ij} (i for pattern number, j for left or right branch), the number of patterns belonging to class C_k is $n_{ij}(k)$.

Entropy of branch $j = Ent(I, f_i, j)$.

The entropy of the system after testing on attribute f_i:

$$Ent(I, f_i) = \sum_{j=1}^{2} \frac{n_{ij}}{\sum_j n_{ij}} Ent(I, f_i, j).$$

4.3 Change in entropy after testing on feature f_i:

$$\Delta Ent(f_i) = Ent(i) - Ent(I, f_i).$$

4.4 Select a feature f_k such that

$$\Delta Ent(f_k) > Ent(f_i)$$

for all $i = 1, 2, \ldots, l, i \neq k$.

4.5 Feature f_k is then the root of the decision tree.

5. *Build the next level of the decision tree.*

Select a feature from the remaining ones, for which the change in entropy is maximum, as the node of next level.

6. Repeat steps 3 through 5 until either all features are exhausted or their number is less than a predetermined threshold or change in entropy is zero.

Here we provide the various entropy measures used in step 3 of the fuzzy ID3 algorithm. Let these be denoted as cases a-g. Note that μ_{ij} is calculated by eqn(2.3.8). Here p_k is the *a priori* probability of the *kth* class and μ_{ij} denotes the member of the jth pattern to the *ith* class. We prune a branch if the number of patterns going down that branch is less than a threshold t.

Case a.

$$\begin{aligned} \textit{Entropy} &= -\sum_{i=1}^{l} \frac{N_i}{N} \log_2 \frac{N_i}{N} - \frac{1}{N} \sum_{i=1}^{l} \sum_{j=1}^{N} [\mu_{ij} \log_2 \mu_{ij} + (1-\mu_{ij}) \log_2 (1-\mu_{ij})] \quad (2.3.9) \\ &= -\sum_{i=1}^{l} p_i \log_2 p_i - \frac{1}{N} \sum_{i=1}^{l} \sum_{j=1}^{N} [\mu_{ij} \log_2 \mu_{ij} + (1-\mu_{ij}) \log_2 (1-\mu_{ij})] \end{aligned}$$

Case b.

Same as Case a, but without pruning.

Case c.

$$\textit{Entropy} = -\frac{1}{N} \sum_{i=1}^{l} (\sum_{j=1}^{N} \mu_{ij}) \log_2 (\sum_{j=1}^{N} \mu_{ij}) \quad (2.3.10)$$

Case d.

$$\textit{Entropy} = -\sum_{i=1}^{l} \frac{N_i}{N} \log_2 \left(\frac{N_i}{N}\right) - \frac{1}{N} \sum_{i=1}^{l} (\sum_{j=1}^{N} \mu_{ij}) \log_2 (\sum_{j=1}^{N} \mu_{ij}) \quad (2.3.11)$$

$$= -\sum_{i=1}^{l} p_i \log_2 p_i - \frac{1}{N}\sum_{i=1}^{l}(\sum_{j=1}^{N}\mu_{ij})\log_2(\sum_{j=1}^{N}\mu_{ij})$$

Case e.

$$\begin{aligned} \textit{Entropy} &= -\frac{1}{N}\sum_{i=1}^{l}(\sum_{j=1}^{N}\mu_{ij})\log_2(\sum_{j=1}^{N}\mu_{ij}) \\ &\quad -\frac{1}{N}\sum_{i=1}^{l}\sum_{j=1}^{N}[\mu_{ij}\log_2\mu_{ij} + (1-\mu_{ij})\log_2(1-\mu_{ij})] \end{aligned} \tag{2.3.12}$$

Case f.

$$\begin{aligned} \textit{Entropy} &= -\sum_{i=1}^{l}\frac{N_i}{N}\log_2\left(\frac{N_i}{N}\right) - \frac{1}{N}\sum_{i=1}^{l}\sum_{j=1}^{N} min(\mu_{ij}, 1-\mu_{ij}) \\ &= -\sum_{i=1}^{l} p_i \log_2 p_i - \frac{1}{N}\sum_{i=1}^{l}\sum_{j=1}^{N} min(\mu_{ij}, 1-\mu_{ij}) \end{aligned} \tag{2.3.13}$$

Case g. Conventional ID3 algorithm

$$\begin{aligned} \textit{Entropy} &= -\sum_{i=1}^{l}\frac{N_i}{N}\log_2\left(\frac{N_i}{N}\right) \\ &= -\sum_{i=1}^{l} p_i \log_2 p_i \end{aligned} \tag{2.3.14}$$

2.4 A Measure for Evaluating the Performance of the Decision Tree

Decision trees generated by different algorithms may vary in size and structure, and this affects the effectiveness of the decision tree and rules extracted from it. So, how do we evaluate the efficiency of a decision tree ? We propose a new measure, named *T-measure*, for evaluating the efficiency of a decision tree keeping in mind the following few points :

- the lesser the depths of the leaf nodes of the tree the better it is since it takes less time to reach to a decision,
- the existence of unresolved leaf is undesirable , and

- the distribution of labeled leaf nodes at different heights affects the performance of the tree; a tree whose frequently accessed leaf nodes are at lower depths is efficient in terms of time.

DEFINITION 2.4.1. *The* **T-measure**, *T, for a decision tree is defined as*

$$T = \frac{2n - \sum_{i=1}^{N_{lnodes}} w_i d_i}{2n - 1} \tag{2.4.15}$$

and

$$w_i = \begin{cases} \frac{N_i}{N} \text{ for a resolved leaf node} \\ \frac{2N_i}{N} \text{ otherwise,} \end{cases} \tag{2.4.16}$$

where n is the number of binary attributes of a pattern, d_i is the depth of a leaf node, N_{lnodes} is the number of leaf nodes, N is the total number of patterns in the training set and N_i is the total number of patterns in the training set that percolate down to the ith leaf node.

The value of T lies in the interval [0,1). A value 0 for T is undesirable and a value close to 1 signifies a good decision tree.
Now, we demostrate the evaluation of the T-measure with an example. Consider a two class problem, with *two – dimensional* patterns, with the two decision trees in Fig. 2.2 from two different algorithms, say.
For the decision tree in Fig. 2.2a

$$\begin{aligned} T &= \frac{2 \times 2 - 0.5 \times 1 - 0.4 \times 2 - 0.2 \times 2}{2 \times 2 - 1} \\ &= 0.77 \end{aligned}$$

and for the decision tree in Fig. 2.2b

$$\begin{aligned} T &= \frac{2 \times 2 - 0.5 \times 2 - 0.4 \times 1 - 0.2 \times 2}{2 \times 2 - 1} \\ &= 0.70. \end{aligned}$$

Here we see that the first decision tree is better than the second one since the fraction of training set patterns in node of depth *one* is more in the first case.

Theorem: *The value of T-measure lies within 0 and 1, i.e.,* $0 \leq T < 1$.

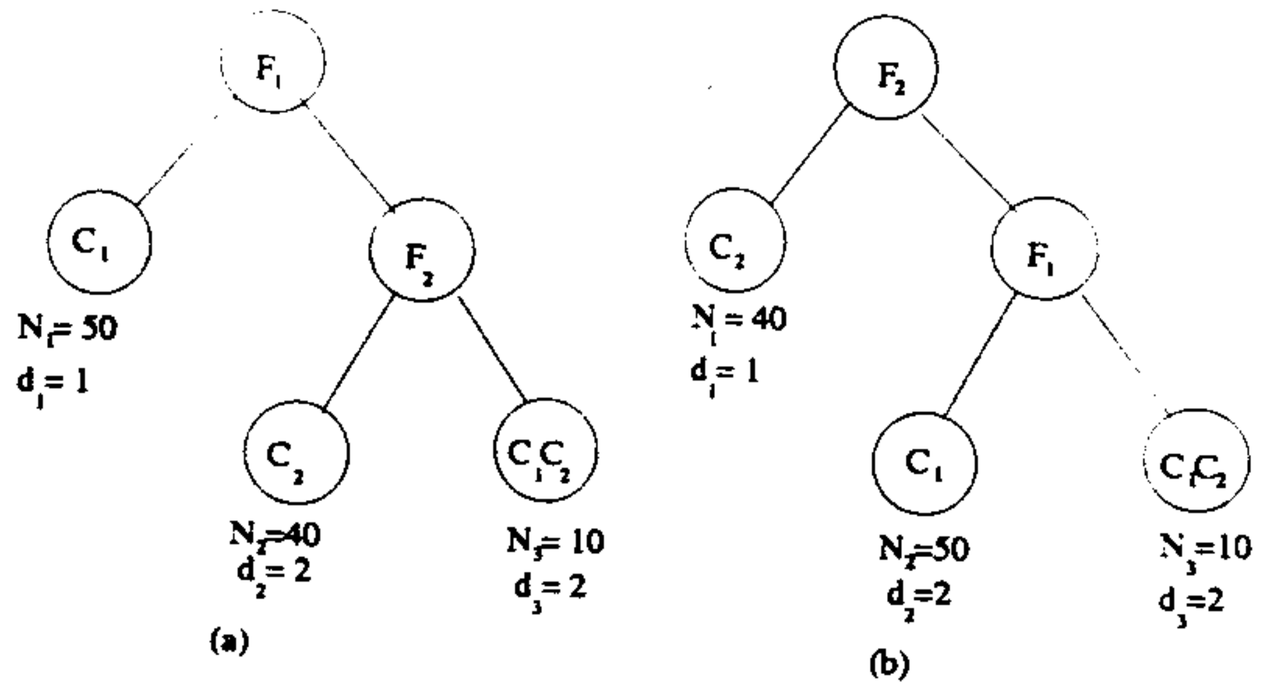

Figure 2.2: Evaluation of T-measure.

Proof:

Let us first establish the upper limit. Since

$$w_i = \begin{cases} \frac{N_i}{N} \text{ for a resolved leaf node} \\ \frac{2N_i}{N} \text{ otherwise} \end{cases}$$

$$w_i \geq \frac{N_i}{N}, i = 1, 2, \ldots N_{lnodes} \tag{2.4.17}$$

and

$$d_i \geq 1, i = 1, 2, \ldots, N_{lnodes}. \tag{2.4.18}$$

Hence

$$\sum_{i=1}^{N_{lnodes}} w_i d_i \geq \sum_{i=1}^{N_{lnodes}} \frac{N_i}{N} \tag{2.4.19}$$

i.e.,

$$\sum_{i=1}^{N_{lnodes}} w_i d_i \geq 1 \tag{2.4.20}$$

since

$$\sum_{i=1}^{N_{lnodes}} N_i = N$$

So,

$$2n - \sum_{i=1}^{N_{lnodes}} w_i d_i < 2n - 1 \tag{2.4.21}$$

i.e.

$$T = \frac{2n - \sum_{i=1}^{N_{lnodes}} w_i d_i}{2n-1} < 1 \tag{2.4.22}$$

Now, we check the lower bound for T. We have

$$w_i \leq \frac{2N_i}{N}, i = 1, 2, \ldots, N_{lnodes} \tag{2.4.23}$$

and

$$d_i \leq n, i = 1, 2, \ldots, N_{lnodes} \tag{2.4.24}$$

So,

$$\sum_{i=1}^{N_{lnodes}} w_i d_i \leq \sum_{i=1}^{N_{lnodes}} \frac{2N_i n}{N} = 2n \tag{2.4.25}$$

i.e.,

$$0 \leq 2n - \sum_{i=1}^{N_{lnodes}} w_i d_i \tag{2.4.26}$$

i.e.

$$0 \leq \frac{2n - \sum_{i=1}^{N_{lnodes}} w_i d_i}{2n-1} = T. \tag{2.4.27}$$

Thus,

$$0 \leq T < 1. \tag{2.4.28}$$

2.5 Results

Here we present some results demonstrating the effectiveness of the revised fuzzy ID3 algorithm on a set of 871 Indian Telugu vowel sounds [20] The vowel sounds, collected by trained personnel, were uttered by three male speakers in the age group of 30 to 35 years, in a Consonant-Vowel-Consonant context. The details of the method are available in [23]. The data set has three features; F_1, F_2 and F_3 corresponding to the first, second and third vowel format frequencies obtained through spectrum analysis of the speech data. Note that the boundaries of the classes in the given data set are seen to be

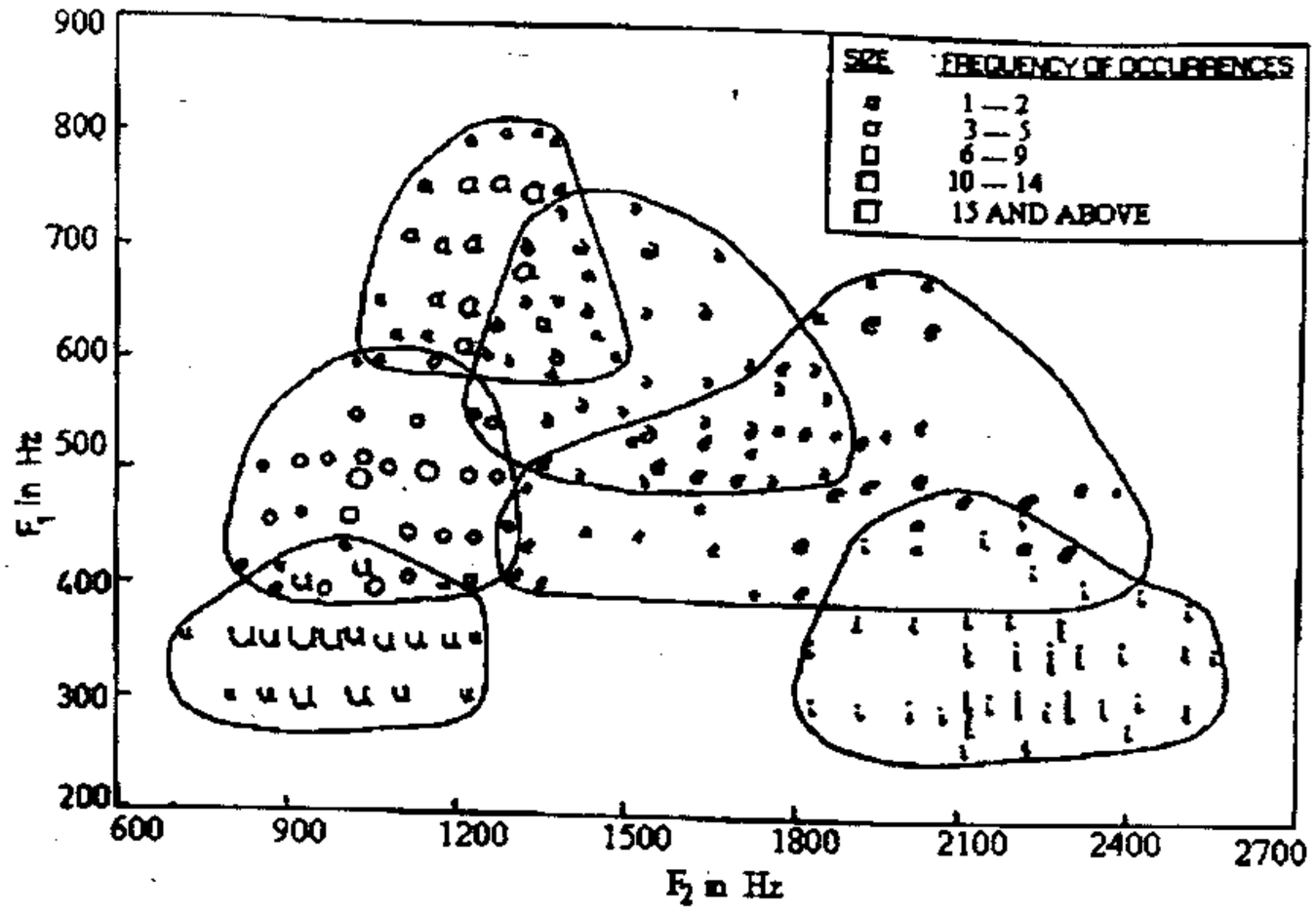

Figure 2.3: Vowel diagram in $F_1 - F_2$ plane

ill-defined (fuzzy). Fig. 2.3 shows a 2D projection of the 3D feature space of the six vowel classes ∂, a, i, u, e, o in the $F_1 - F_2$ plane, for ease of depiction. The recognition scores (%) and T-values for the different models of the fuzzy ID3 models, for both training and test sets, are given below. The results in the table shows that the Fuzzy ID3 with the Entropy measure a gives the best performance.

PERFORMANCE ON TRAINING SET									
Model	Train size (%)	Score (%)							T-measure
		1	2	3	4	5	6	Total	
a	10	79.24	69.91	89.00	83.59	81.48	72.17	81.09	0.70
	20	84.73	66.89	89.65	87.59	70.32	65.63	77.56	0.69
	30	77.66	61.67	91.41	87.26	69.67	63.50	75.91	0.68
	40	78.03	56.15	90.12	86.88	69.37	63.18	74.73	0.67
	50	79.66	57.29	94.19	91.08	61.76	64.26	74.91	0.67
b	10	89.34	64.45	89.83	87.39	72.34	60.65	77.24	0.53
	20	90.25	53.42	91.95	87.88	68.51	52.87	73.46	0.53
	30	88.56	62.51	94.08	89.60	54.98	43.39	69.89	0.53
	40	91.15	53.62	95.45	92.55	46.63	31.79	65.60	0.53
	50	90.08	52.55	94.86	91.56	41.84	25.93	62.97	0.53
c	10	87.15	71.07	91.31	84.26	73.87	70.03	79.45	0.70
	20	82.39	57.84	89.11	87.79	73.68	62.17	76.11	0.69
	30	79.01	62.44	91.38	86.38	65.96	62.84	74.79	0.70
	40	76.46	55.49	91.85	86.46	65.52	63.10	74.05	0.69
	50	79.85	51.28	93.14	87.80	61.18	62.88	73.17	0.67
d	10	83.22	67.90	82.65	80.55	77.17	62.23	76.87	0.66
	20	80.82	65.74	93.78	89.90	66.75	61.72	76.35	0.70
	30	77.81	60.22	94.22	90.90	63.23	65.40	75.75	0.69
	40	77.50	52.84	94.60	91.01	62.55	65.31	74.80	0.68
	50	77.50	55.50	94.06	90.02	59.86	64.05	73.88	0.68
e	10	83.41	58.70	87.77	85.98	78.74	70.19	79.02	0.71
	20	82.32	62.42	88.47	80.28	71.72	68.74	76.28	0.71
	30	76.85	57.65	89.47	82.65	68.19	63.97	74.15	0.69
	40	79.79	55.84	94.22	90.16	60.65	65.58	74.49	0.69
	50	74.99	55.51	94.52	89.34	59.92	66.05	74.05	0.69
f	10	31.50	12.86	78.29	71.02	69.70	77.38	65.98	0.64
	20	43.44	15.44	80.77	68.32	54.29	75.26	61.32	0.61
	30	38.00	9.06	80.60	68.06	44.81	74.97	58.12	0.60
	40	31.09	16.27	82.73	70.02	39.38	79.34	58.89	0.60
	50	27.06	14.67	82.97	70.87	41.30	80.06	58.84	0.60
g	10	28.39	15.45	79.25	65.76	74.29	74.52	63.25	0.64
	20	17.03	16.89	79.40	54.22	61.54	73.25	57.72	0.61
	30	13.54	6.08	82.82	60.72	52.34	77.69	57.05	0.60
	40	14.39	7.70	78.36	61.32	56.72	76.62	57.24	0.60
	50	14.30	7.92	79.58	60.94	52.69	76.34	56.48	0.59

Table 2.1 Performance of the fuzzy ID3 on training set.

PERFORMANCE ON TEST SET								
Model	Train size (%)	Score (%)						
		1	2	3	4	5	6	Total
a	10	65.43	50.31	85.35	82.89	70.91	68.70	72.75
	20	69.42	59.26	88.81	79.79	67.81	61.96	71.87
	30	72.53	59.00	87.89	84.70	63.23	65.35	72.55
	40	74.80	49.19	91.75	89.01	59.26	58.82	71.01
	50	73.06	59.97	91.34	89.36	58.77	66.53	73.14
b	10	68.63	50.30	82.87	84.84	68.26	60.73	70.66
	20	71.45	53.92	88.20	84.28	62.14	51.82	68.83
	30	74.20	54.23	87.95	85.05	53.08	39.42	64.52
	40	81.60	55.43	92.37	88.70	46.89	28.28	62.84
	50	82.42	55.26	93.65	91.95	40.29	25.47	61.62
c	10	61.74	51.68	87.00	82.38	67.21	61.59	70.41
	20	65.12	58.00	89.40	81.53	63.48	57.61	70.04
	30	69.80	54.57	90.01	84.99	59.26	60.93	70.37
	40	72.02	48.05	91.90	89.37	59.48	57.37	70.62
	50	70.05	54.88	90.01	84.63	64.50	68.33	73.09
d	10	66.91	52.84	90.07	87.37	57.94	55.60	68.96
	20	70.25	53.01	90.49	89.42	61.89	55.15	70.49
	30	67.29	53.97	89.97	85.65	61.17	58.88	70.41
	40	66.94	50.67	92.18	88.30	61.87	63.65	71.85
	50	67.17	53.76	92.46	88.09	61.90	67.88	73.25
e	10	65.19	47.28	87.03	86.79	66.11	59.04	70.27
	20	71.45	51.50	86.92	84.50	66.17	62.07	71.49
	30	70.97	56.71	90.82	84.73	57.23	60.87	70.34
	40	72.31	57.45	89.72	84.84	60.53	60.85	71.07
	50	72.92	50.60	91.70	86.82	56.82	61.11	70.29
f	10	29.21	27.53	77.14	74.04	64.31	72.99	63.57
	20	32.67	23.89	87.68	71.10	50.05	85.19	64.47
	30	41.45	20.41	82.46	72.79	44.33	85.05	62.28
	40	30.85	22.16	81.53	72.59	38.47	86.94	60.42
	50	31.02	18.46	81.08	72.09	39.13	88.22	60.65
g	10	13.20	24.50	68.67	63.73	61.01	48.78	52.69
	20	19.65	23.46	68.15	65.41	51.00	51.40	51.42
	30	10.81	19.75	76.44	72.14	47.06	32.20	48.08
	40	9.66	21.93	76.62	72.63	44.82	34.46	48.39
	50	5.32	18.06	74.52	72.75	39.81	25.53	44.54

Table 2.2 Performance of the fuzzy ID3 on test set.

Chapter 3

Rule Generation

3.1 Algorithm

To generate the architecture of a neural network and to encode it with *a priori* knowledge, we need to know the rules extracted by the fuzzy ID3. The path from the root to a leaf can be traversed to generate the corresponding rule for a class labeled by that leaf node. In this manner one extracts a set of rules corresponding to different pattern classes in the conjunctive form of the attributes or features encountered along the path. The ith feature is marked as A_i or $\overline{A_i}$ depending on whether the traversal is made along the right or left branch of the node in the decision tree. Each rule is also marked by its frequency (number of patterns)and the class(es) it corresponds to.

Algorithm 3.1: Algorithm for Rule Generation

```
for each path from root node to leaf do
    begin
        rule = φ;
        current_node = root_node;
        do while (current_node ≠ root_node)
            begin
                if the leaf is in the left subtree and node has decisive feature F_i
                    rule = rule ∧ f̄_i
            else
                    rule = rule ∧ f_i
end
```

assign the decision of the rule by the label of the leaf node.

assign the frequency of the rule by the number of patterns in the leaf node.

end.

The scheme of extracting the rules from the decision tree is shown below with an example. Suppose the training set consists of 21 patterns, from three pattern classes, with three features f_1,f_2 and f_3. After splitting each feature into the three linguistic variables *low, medium and high* resulting in *nine – dimensional* symbolic features, *viz.*, $L_1, M_1, H_1, L_2, M_2, H_2, L_3, M_3, H_3$ the decision tree has been constructed as the one depicted below.

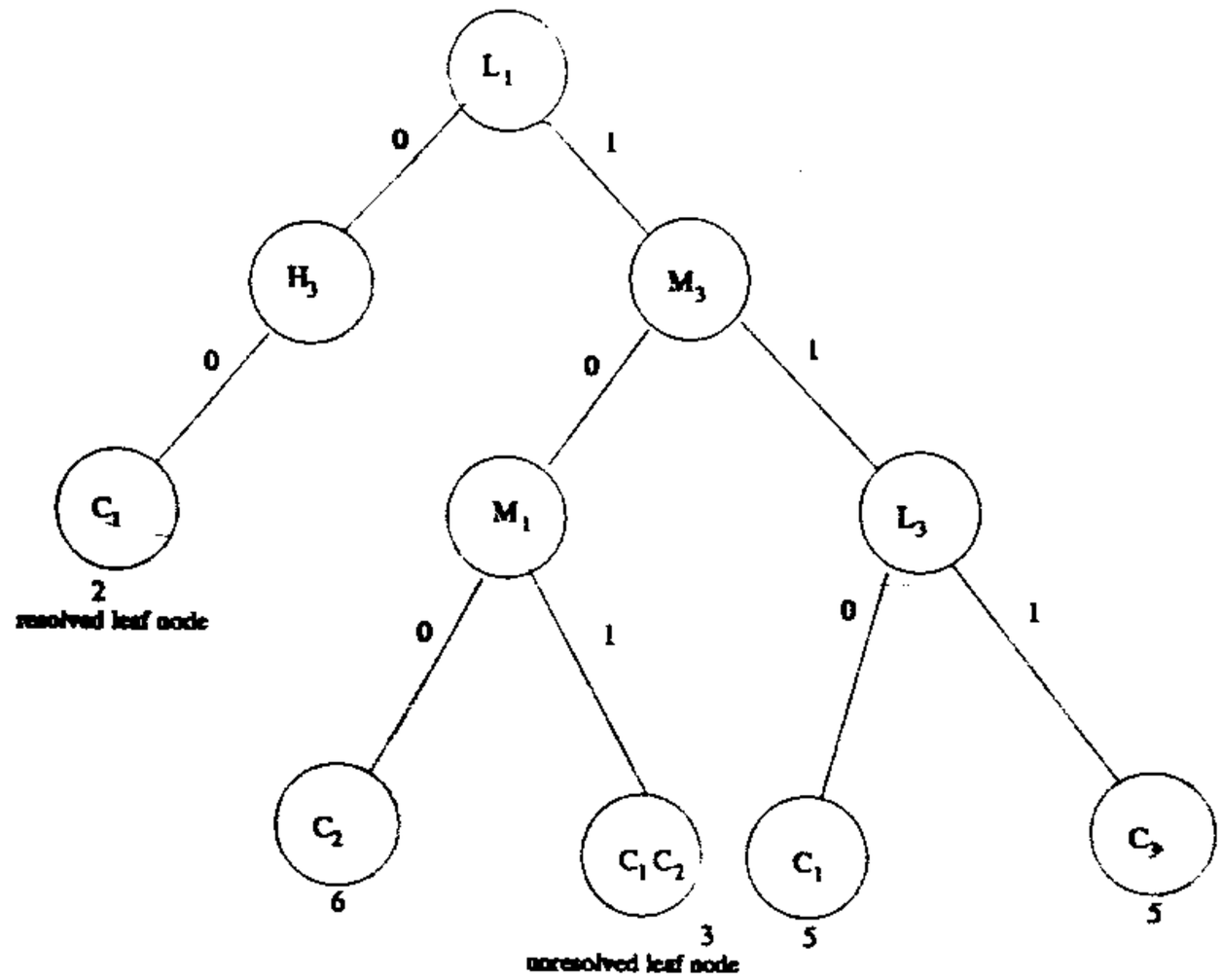

Figure 3.1: An example of a decision tree.

By the above algorithm the following rules are extracted:

1. $\overline{L}_1 \wedge \overline{H}_3 \rightarrow C_1 \quad 2;$
2. $L_1 \wedge \overline{M}_3 \wedge \overline{M}_1 \rightarrow C_2 \quad 6;$
3. $L_1 \wedge \overline{M}_3 \wedge M_1 \rightarrow C_1, C_2 \quad 3;$
4. $L_1 \wedge M_3 \wedge \overline{L}_3 \rightarrow C_1 \quad 5;$
5. $L_1 \wedge M_3 \wedge L_3 \rightarrow C_3 \quad 5;$

3.2 Quantitative Measures for Performance Evaluation

Some quantitative measures have been calculated for the rules extracted above to compare their performances. A new measure, which is denoted here as *Coverage*, has been introduced here. We provide these measures below:

Let N be an $l \times l$ matrix whose $(i,j)th$ element n_{ij} indicates the number of patterns actually belonging to class i but classified as class j.

DEFINITION 3.2.1. **Accuracy**: *It is the correct classification percentage, provided by the rules on a test set defined as* $\frac{n_{ic}}{n_i}$, *where* n_i *is equal to the number of points in class* i *and* n_{ic} *of these points are correctly classified.*

DEFINITION 3.2.2. **User's Accuracy**: *If* n'_i *points are found to be classified into class* i, *then user's accuracy* (U) *is defined as* $U = \frac{n_{ic}}{n'_i}$. *This gives a measure of the confidence that a classifier attributes to a region as belonging to a class. In other words, it denotes the level of purity associated with a region.*

DEFINITION 3.2.3. **Kappa** *[24]: The coefficient of agreement called "kappa" measures the relationship of beyond chance agreement to expected disagreement. It uses all the cells in the confusion matrix, not just the diagonal elements. The estimate of kappa* (K) *is the proportion of agreement after chance agreement is removed from consideration. The kappa value for class* i (K_i) *is defined as*

$$K_i = \frac{n.n_{ic} - n_i.n'_i}{n.n'_i - n_i.n'_i} \tag{3.2.1}$$

The numerator and denominator of overall kappa are obtained by summing the respective numerators and denominators of K_i *seperately over all classes.*

DEFINITION 3.2.4. **Fidelity** : *This represents how closely the rule base approximates the parent decision tree model. This is measured as the percentage of the test set for which tree and the rule base output agree. Note that fidelity may or may not be greater than accuracy.*

DEFINITION 3.2.5. **Confusion** *[24] : This measure quantifies the goal that the "Confusion should be restricted within minimum number of classes". This property is helpful in higher level decision making. Let* $\hat{n_{ij}}$ *be the mean of all* n_{ij} *for* $i \neq j$. *Then we define*

$$Conf = \frac{Card\{n_{ij} : n_{ij} \geq \hat{n}_{ij}, i \neq j\}}{l} \tag{3.2.2}$$

for an l *class problem. The lower the value of Conf, the lesser is the number of classes between which confusion occurs.*

DEFINITION 3.2.6. **Coverage**: *We define it as the ratio between the total number of patterns associated with the rules corresponding to resolved nodes to the total number of patterns in all the rules and hence the leaf nodes.*

In a training set where the rules can perfectly classify all the patterns this value is 1. and if they cannot classify any pattern then it is 0. For example, in the above example the measure of coverage is.

$$Coverage = \frac{2+6+5+5}{2+6+3+5+5} = \frac{18}{21}$$

3.3 Evaluation of Rules

The above measures are evaluated for the rules extracted by models a, b, f (section 2.3)using the vowel data.

Model	Train size(%)	Accuracy(%)	User Accuracy(%)	Kappa	Confusion	Coverage	Fidelity
a	10	63.20	72.67	0.67	2.30	0.80	85.57
	20	62.70	73.15	0.67	2.37	0.78	84.69
	30	59.74	75.47	0.70	2.37	0.72	81.37
	40	59.62	77.14	0.72	2.20	0.73	78.91
	50	60.06	75.40	0.70	2.34	0.75	75.51
b	10	60.33	71.45	0.65	2.54	0.76	75.72
	20	60.14	73.58	0.67	2.24	0.79	75.95
	30	60.52	77.63	0.72	2.26	0.73	77.57
	40	59.46	79.36	0.75	2.38	0.69	74.12
	50	60.18	80.65	0.77	2.06	0.72	73.05
f	10	60.69	77.22	0.72	2.19	0.71	46.83
	20	59.70	71.80	0.67	1.89	0.60	52.26
	30	53.59	74.64	0.70	1.93	0.54	49.38
	40	53.39	73.95	0.69	1.99	0.52	47.33
	50	53.27	72.71	0.68	1.92	0.52	44.63

Table 3.1. Quantitative measures for the rules.

Chapter 4

Mapping of Rules to Neural Network Architecture

4.1 Multilayer Perceptron (MLP)

A multilayer perceptron is one of the most powerful ANN models for pattern classification. An MLP is represented by a labeled directed graph where each node can perform some calculations; and each connection or link conveys a signal determined by the weight of the link.

Consider Fig. 4.1.The output of a neuron in a layer h of a multilayer perceptron (MLP), except the input layer (where $h = 0$) is

$$y_j^h = \frac{1}{1 + \exp\left[-(net_j + \theta^h)/\theta_0\right]} \tag{4.1.1}$$

where

$$net_j = \sum_i y_i^{h-1} w_{ji}^{h-1}, \tag{4.1.2}$$

θ^h serves as a threshold or bias, y_j^{h-1} is the state of the ith neuron in the preceding $(h-1)th$ layer, w_{ji}^{h-1} is the weight connection from the ith neuron in $(h-1)th$ layer to jth node in the hth layer.

Before a neural network can be used for pattern classification, it must be trained to adjust the weights. The back-propagation algorithm is central to much work on learning neural networks and has been used in this dissertation work.

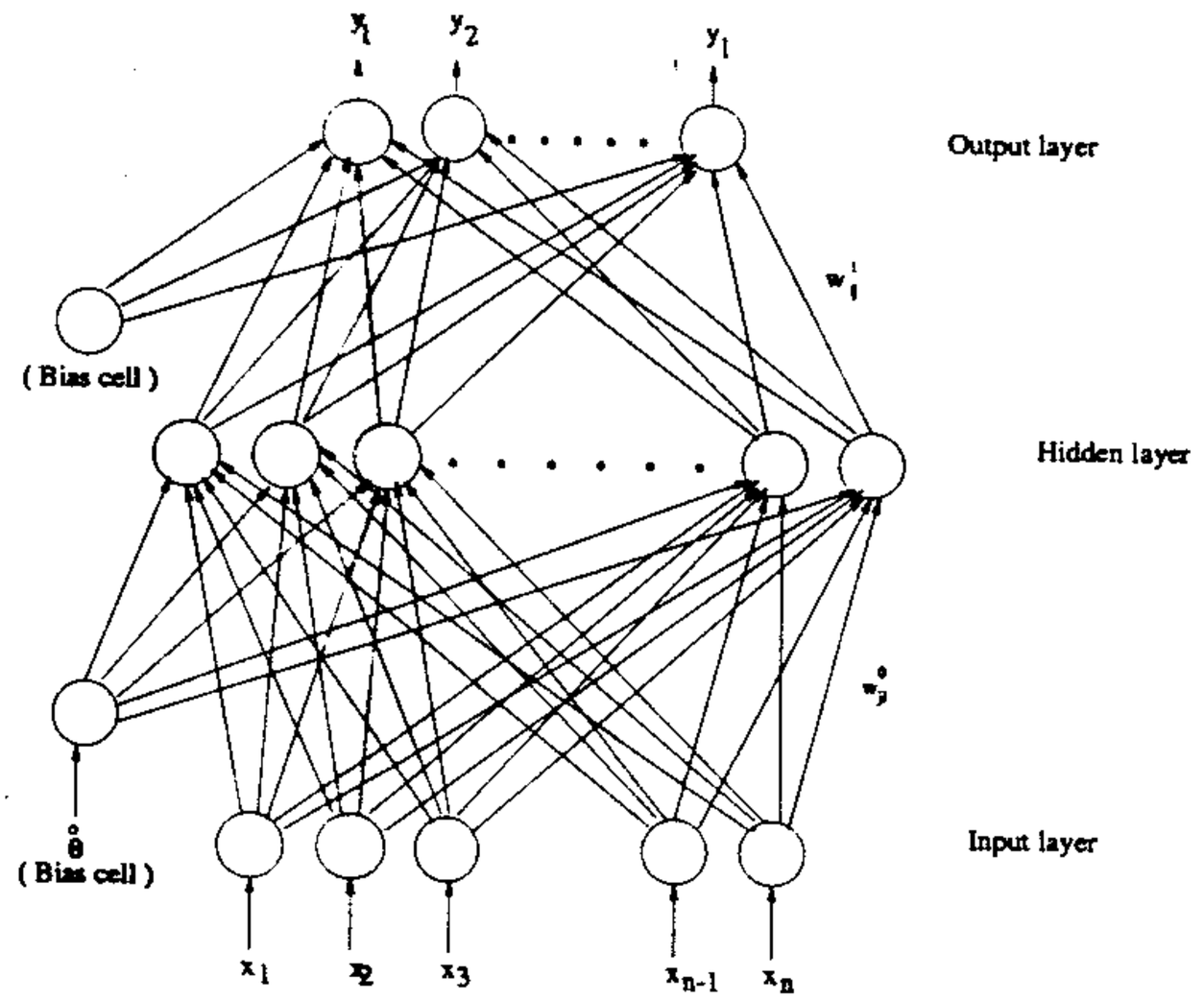

Figure 4.1: A Multilayer Perceptron (MLP).

Let $\{(\mathbf{x}^1, \mathbf{d}^1), (\mathbf{x}^2, \mathbf{d}^2), \ldots, (\mathbf{x}^{(p)}, \mathbf{d}^{(p)})\}$ be a set of p training patterns (input-output pairs), where $\mathbf{x}^{(i)} \in \mathcal{R}^n$ is the input vector in the $n-$*dimensional* pattern space, and $\mathbf{d}^{(i)} \in [0,1]^e$, an $l-$*dimensional* hypercube. For classification purpose, l is the number of classes. The squared error cost function most frequently used in the ANN literature is defined as

$$E = \frac{1}{2}\sum_{i=1}^{p} ||\mathbf{y}^{(i)} - \mathbf{d}^{(i)}||^2 \tag{4.1.3}$$

The back propagation algorithm is a gradient descent method to minimize the squared error cost function in the above equation. Backpropagation algorithm is given below :

Algorithm 4.1:Back-propagation algorithm

1. Initialize the weights to small random values.

2. Randomly choose an input pattern $\mathbf{x}^{(\mu)}$.

3. Propagate the signal forward through the network.

4. Compute δ_i^H in the output layer ($o_i = y_i^H$).

$$\delta_i^h = f'(g_i^h)\left[d_i^u - y_i^H\right] \tag{4.1.4}$$

where h_i^l represents the net input to the ith unit in the lth layer and f' is the derivative of the activation function f.

5. Compute the δs for the preceding layers by propagating the errors backwards;

$$\delta_i^h = f'(g_i^h) \sum_j w_{ij}^{h+1} \delta_j^{h+1} \tag{4.1.5}$$

for $h = (H-1), \ldots, 1$.

6. Update weights using

$$\Delta w_{ij}^h = \eta \delta_i^h y_j^{h-1} \tag{4.1.6}$$

7. Go to step 2 and repeat for the next pattern until the error in the output layer is below a pre-specified threshold or a maximum number of iterations is reached.

4.2 Mapping of Rules

Now we discuss the details of the knowledge encoding scheme, used in this work, employing the rules extracted from the decision trees. Let r_{ki} be the ith rule for class C_k with frequency f'_{ki}. Rule involving only one class are selected and the rules corresponding to unresolved nodes of the decision tree are discarded. If there are two rules for a single class C_k the rule involving the highest frequency is considered. The rules from Fig. 3.1 are considered here. they reduce to

1. $L_1 \wedge \overline{M}_3 \wedge \overline{M}_1 \rightarrow \ C_2 \ \ 6;$
2. $L_1 \wedge M_3 \wedge \overline{L}_3 \rightarrow \ C_1 \ \ 5;$
3. $L_1 \wedge M_3 \wedge L_3 \rightarrow \ C_3 \ 5;$

Let r_{ki} be the ith rule for class C_k with frequency f'_{ki}. each rule is mapped using a single hidden node, in order to model the conjunction between the attributes in the rule. Here we compare the results of three methods of mapping the rules.

4.2.1 Model I

The weight w_{ki}^1, between output node k (class C_k) and hidden node i (rule r_{ki}), is set at $f'_{ki} + \epsilon$, where ϵ is a small random number; and $f'_{ki} = 1$. The weight $w_{iA_j}^0$ between attribute

A_j and hidden node i is clamped to $\frac{w^1_{ki}}{Card(r_{ki})} + \epsilon$. Here $Card(r_{ki})$ indicates the number of features/attributes encountered along the traversal path from the root to the leaf containing the pattern corresponding to rule r_{ki} a of class C_k. An example illustrating this scheme is provided in Fig. 4.2 for class C_1.

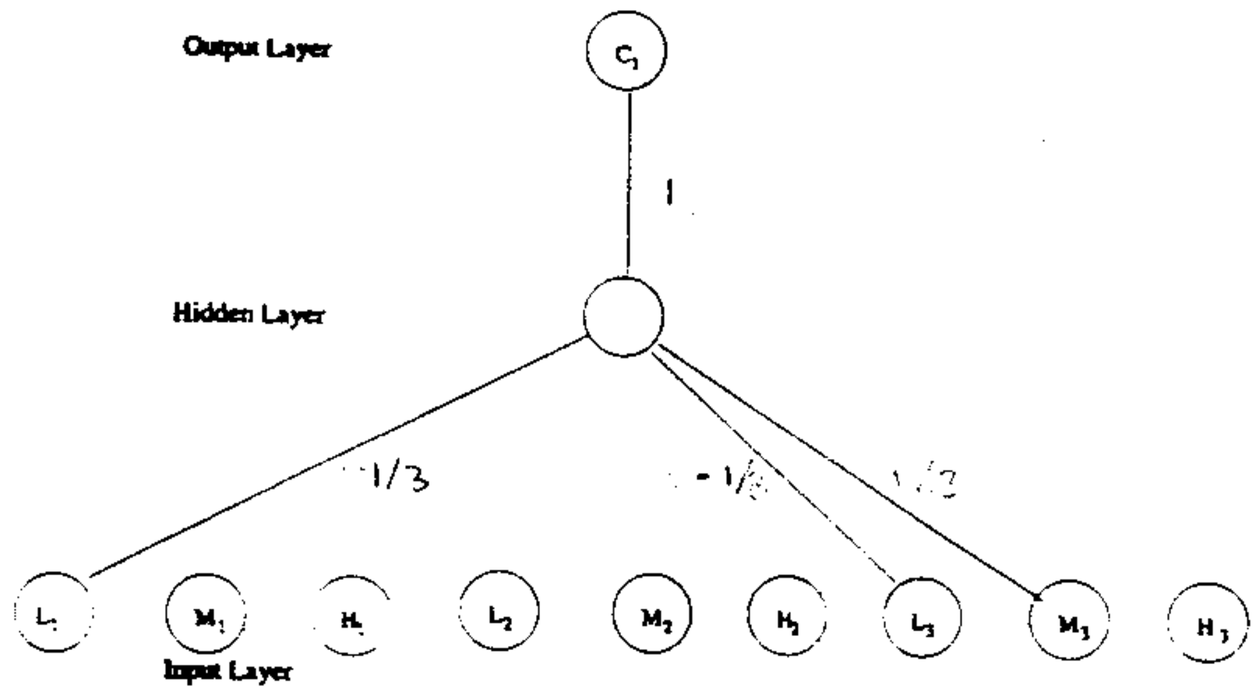

Figure 4.2: Weight encoding using Model I.

4.2.2 Model II

Here a factor $K = \frac{f'_{ki}}{\sum_k f'_{ki}}$ is used to indicate the importance of a rule for a particular class C_k among all rules determining the whole network. The scheme for mapping weight $w^0_{iA_j}$ is the same as in Model I. An example illustrating this scheme is provided in Fig. 4.3.

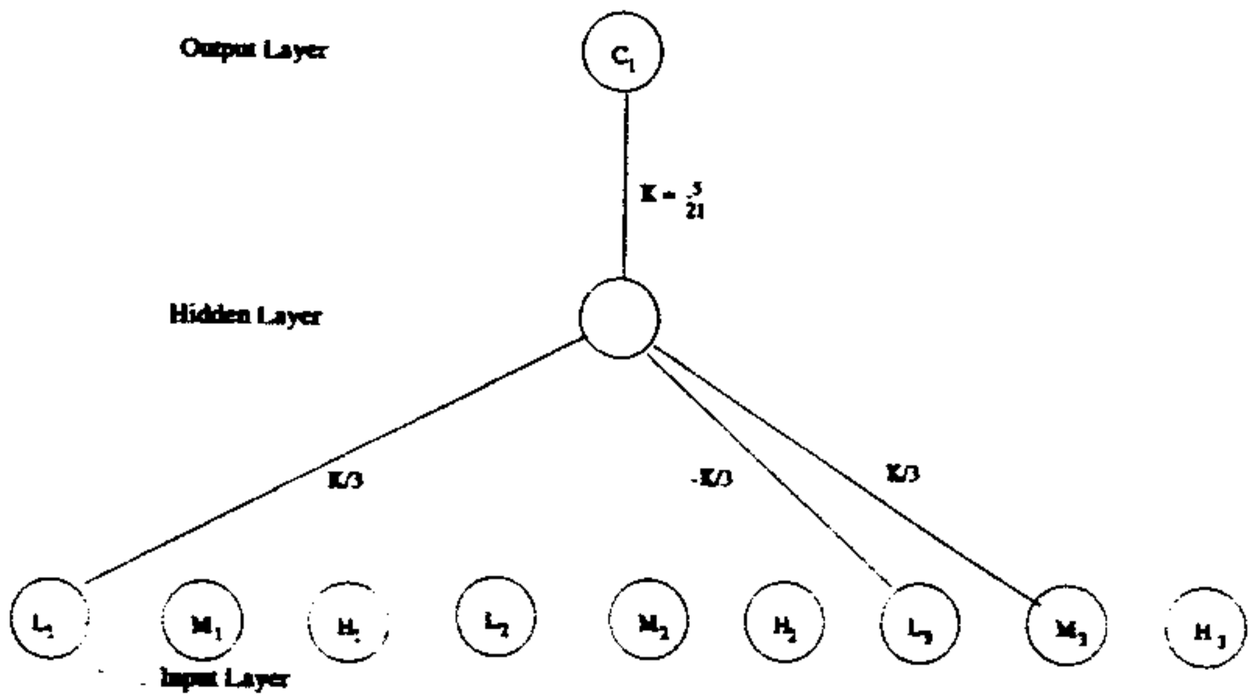

Figure 4.3: Weight encoding using Model II.

4.2.3 Model III

Here, as in Model II, a factor $K = \frac{f'_{ki}}{\sum_k f'_{ki}}$ is used to indicate the importance of a rule for a particular class C_k among all rules determining the whole network. But the scheme for mapping weight $w^0_{iA_j}$ depends on the importance of feature A_j in the corresponding decision tree. While constructing the decision tree the feature associated with a node is chosen on the basis of maximum information gain. Hence the attributes/features appearing in the beginning should be given more weightage in the descending order. Consider Fig. 3.1. We note that the attributes are selected in the order L_1, M_3, L_3 for class C_1. So the weight $w^0_{iA_j}$ is assigned with $\frac{2(Card(r_{ki})-i+1)}{Card(r_{ki})(Card(r_{ki})+1)}K$. It is to be noted that

$$\sum_{i=1}^{Card(r_{ki})} \left| \frac{2(Card(r_{ki}) - i + 1)}{Card(r_{ki})(Card(r_{ki}) + 1)} \right| K \;=\; K \tag{4.2.7}$$

A salient feature of this method is not allowing the initially nonexistent hidden output layer (having very small absolute values) to grow while the training goes on. In these models the number nodes in the hidden layer is equal to the number of classes.An example illustrating this scheme is provided in Fig. 4.4.

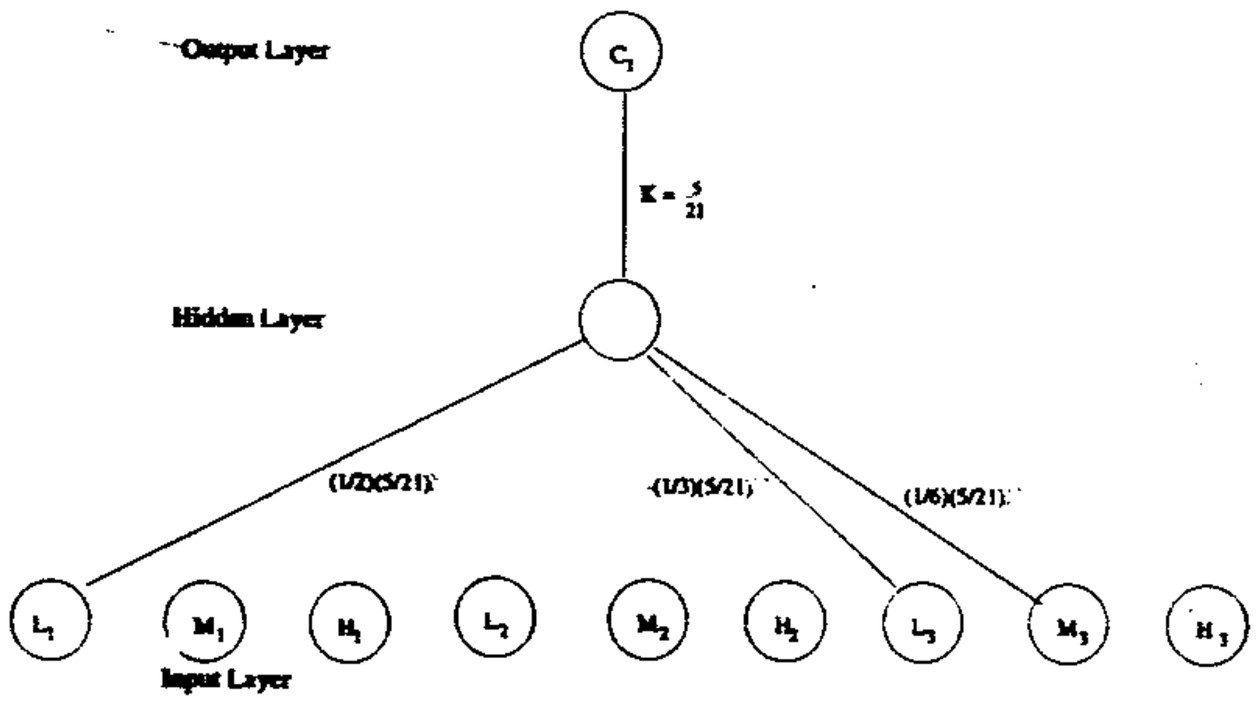

Figure 4.4: Weight encoding using Model III.

4.3 Comparing the Performance of the Neural Networks

A comparison of the performance of the ordinary neural network and the three knowledge-encoded models have been made on the the vowel data as given in the tables below.

Models	Data	Train size (%)	Score (%)							No of Links
			1	2	3	4	5	6	Total	
M L P	TRAINING SET	10	72.17	98.21	89.13	89.30	99.39	98.12	93.33	90.45
		20	56.57	91.40	78.22	84.08	98.12	98.02	87.50	91.20
		30	38.60	90.79	66.23	75.74	97.74	98.79	82.61	91.35
		40	23.75	89.19	62.50	60.81	96.49	97.74	76.81	91.80
	TEST SET	10	35.93	86.93	80.95	82.74	81.39	84.49	78.88	-
		20	31.78	86.54	74.56	74.97	89.75	92.69	79.61	-
		30	28.37	85.31	68.22	75.01	94.34	94.10	79.44	-
		40	22.84	87.01	65.11	66.65	94.96	96.35	77.64	-
Model I	TRAIN SET	10	56.96	86.59	85.25	91.02	97.59	97.92	89.43	65.60
		20	42.65	88.85	79.03	81.23	96.31	97.85	85.63	65.60
		30	24.92	89.50	76.32	78.14	95.49	97.25	82.99	65.20
		40	14.94	87.44	73.97	75.93	96.30	98.94	81.47	67.70
	TEST SET	10	27.69	84.38	80.40	80.55	85.27	87.03	78.95	-
		20	25.16	84.60	76.98	78.21	90.19	91.86	79.70	-
		30	18.78	83.02	75.25	75.46	92.95	95.27	79.56	-
		40	10.10	82.75	73.78	73.33	93.44	97.94	78.88	-
Model II	TRAIN SET	10	11.23	80.56	74.51	71.37	92.22	98.30	78.30	66.50
		20	49.19	90.59	84.72	90.44	97.59	97.92	88.85	65.20
		30	16.47	89.23	77.06	78.70	95.40	97.62	82.70	63.10
		40	3.38	85.16	71.57	73.88	92.69	98.81	78.56	66.10
	TEST SET	10	27.84	84.13	80.40	80.63	84.79	86.15	78.65	-
		20	25.85	84.45	77.04	78.23	90.02	91.87	79.72	-
		30	17.97	83.16	75.25	74.99	93.02	95.19	79.43	-
		40	9.35	81.64	73.69	73.32	93.17	98.23	78.68	-
Model III	TRAINING SET	10	51.74	91.96	85.91	86.59	97.29	94.28	88.68	63.60
		20	37.65	91.58	80.53	84.69	95.67	98.04	86.03	68.25
		30	25.90	87.59	75.49	77.46	96.33	98.46	82.91	66.50
		40	22.81	82.96	75.26	75.13	94.88	97.79	80.76	64.15
	TEST SET	10	27.98	86.20	78.50	83.56	84.11	84.55	78.49	-
		20	25.99	86.03	77.17	77.37	90.57	93.27	80.30	-
		30	18.57	84.47	74.10	76.71	92.89	96.42	80.16	-
		40	17.68	84.67	73.31	73.64	92.53	95.63	79.01	-

Table 4.1. Comparision of performance of knowledge-encoded models with MLP.

Chapter 5

Conclusion

The work done in this dissertation is two-fold and has been tested using the vowel data. The first part involved developing a fuzzy version of ID3 to gain higher accuracy in classification by minimizing noise. The decision tree so constructed is optimize by elimination of noise by appropriate choice of linguistic functions and followed by pruning. However, methods of finding a good fuzzy entropy and keeping a good balance with its classical counter part, say, Shannon's entropy, is not known. In this work, this aspect has been analysed by proposing a few entropy functions. It was observed that the elimination of noise while constructing the decision tree plays an major role in improving the performance of the algorthim over the previous ones it is based upon.

A measure denoted here as $T - measure$, for evaluating the performance of the decision tree is proposed.This measure considers both the accuracy of classification and time complexity of the decision tree while making a decision.

The next step involves extracting the rules from the decision trees. A new measure, denoted here as *Coverage*, has been introduced to access the amount of data the rules cover.

Finally, a new way to map the rules into the neural network has been proposed. It gives higher weightage to the attributes in descending order of their appearance in the rules. The network so generated has 1-2% improvement over the ordinary neural network with similar number of layers and number nodes in each layer. However, the number of links in the generated (knowledge encoded) network is about 30% less than that in the ordinary neural network.

Bibliography

[1] B. D. Ripley, *Pattern Recognition and Neural Networks*, New York: Cambridge University Press,1996.

[2] P. Lippmann, "An Introducion to computing with neural nets" *IEEE Acoustics, Speech and Signal Processing Magazine*, vol 44, pp. 4-22, 1987.

[3] B. Irie and S. Miyake, "Capabilities of three-layered perceptrons", in *Proc. IEEE Int. Conf. Neural Networks*, pp I-641–I-648, 1988.

[4] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," *Advances in Neural Information Processing Systems*, Morgan-Kaufmann, Los Altos, CA, pp 524–532, 1990.

[5] R. Reed, "Pruning algorithms - a survey," *IEEE Transactions on Neural Network*, vol. 4, No. 5, pp. 740–747, 1993.

[6] J. R. Quinlan, "Learning efficient classification procedures and their applications to chess end-games," in *Machine Learning*, R. S. Michalski, J.G. Carbonell and T. M. Mitchell, Eds., Los Altos, CA: Morgan Kaufmann, 1983.

[7] T. G. Dietterich, H. Hild and G. Bakiri, "A comparative study of ID3 and back-propagation for English text-to-speech mapping," in *Proc. Seventh Int. Conf. Machine Learning*, (Austin, TX), 1990.

[8] D. H. Fisher and K. B. Mckusick, "An empirical comparison of ID3 and back-propagation," in *Proc. Eleventh Int. Joint Conf. Artificial Intelligence*, pp. 788-793, 1989.

[9] R. Setiono and W. K. Leow, "On mapping decision trees and neural networks," *Knowledge-Based Systems*, vol. 12, pp. 95-99, 1999.

[10] L. M. Fu, "Knowledge-based connectionism for revising domain theories," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 23, pp. 173–182, 1993.

[11] G. G. Towell and J. W. Shavlik, "Knowledge-based artificial neural networks," *Artificial Intelligence*, vol. 70, pp. 119–165, 1994.

[12] M. Banerjee, S. Mitra and S. K. Pal, "Rough fuzzy MLP: Knowledge encoding and classification," *IEEE Transactions on Neural Networks*, vol. 9, pp. 1203-1216, 1998.

[13] Y. H. Pao, *Adaptive Pattern-Recognition and Neural Network*, Addison-Wesley, Reading, MA, 1989.

[14] J. R. Quinlan, "Induction on Decision Trees," in *Machine Learning*, vol 1, pp. 81-106, 1986.

[15] L. A. Zadeh, "Fuzzy Sets," *Information and Control*, vol. 15, pp. 338-353, 1965.

[16] J. C. Bezdek and S. K. Pal, eds., *Fuzzy Models for Pattern Recognition: Methods that Search Structures in Data*, New York: IEEE Press, 1992.

[17] H. Ichihashi, T. Shirai, K. Nagasaka and T. Miyoshi, "Neuro-Fuzzy ID3: A method of inducing fuzzy decision trees with linear programming for maximizing entropy and algebraic methods," *Fuzzy Sets and Systems*, vol. 81, no. 1, pp. 157-167, 1996.

[18] N. R. Pal, S. Chakraborty and A. Bagchi, "RID3 like algorithm for real data," *Information Sciences*, vol. 96, pp. 271-290, 1997.

[19] P. K. Singal, S. Mitra and S. K. Pal, "Incorporation of fuzziness in ID3 and generation of network architecture," *Neural Computing and Applications*. (Submitted).

[20] S. K. Pal and S. Mitra, "Multilayer Perceptron, Fuzzy Sets, and Classification," *IEEE Transactions on Neural Networks*, vol. 3, no. 5, September 1992.

[21] S. K. Pal and D. Dutta Majumdar, *Fuzzy Mathematical Approach to Pattern Recognition*, New York: Wiley (Halsted Press), 1986.

[22] S. K. Pal and S. Mitra, "Neuro-fuzzy Pattern Recognition: Methods in Soft Computing," John Wiley, New York, 1999.

[23] S. K. Pal and P. K. Pramianik, "Fuzzy measures in determining seed points in clustering," *Pattern Recognition Letters*,", vol 4, pp. 159-164, 1986.

[24] S. K. Pal, S. Mitra and P. Mitra, "Rough Fuzzy MLP: Modular evolution, rule generation and evaluation," *IEEE Transactions on Knowledge and Data Engineering.* (Submitted)